Bounding the Search Space of the Population Harvest Cutting Problem with Multiple Size Stock Selection

Laura Climent, Barry O'Sullivan, and Steven D. Prestwich

Insight Centre for Data Analytics
Department of Computer Science, University College Cork, Ireland
{laura.climent|barry.osullivan|steven.prestwich}@insight-centre.org

Abstract. In this paper we deal with a variant of the Multiple Stock Size Cutting Stock Problem (MSSCSP) arising from population harvesting, in which some sets of large pieces of raw material (of different shapes) must be cut following certain *patterns* to meet customer demands of certain product types. The main extra difficulty of this variant of the MSSCSP lies in the fact that the available patterns are not known *a priori*. Instead, a given complex algorithm maps a vector of continuous variables called a *values vector* into a vector of total amounts of products, which we call a *global products pattern*. Modeling and solving this MSSCSP is not straightforward since the number of value vectors is infinite and the mapping algorithm consumes a significant amount of time, which precludes complete pattern enumeration. For this reason a representative sample of global products patterns must be selected. We propose an approach to bounding the search space of the values vector and an algorithm for performing an exhaustive sampling using such bounds. Our approach has been evaluated with real data provided by an industry partner.

1 Introduction

The *Cutting Stock Problem* (CSP) [6] is a well-known NP-hard optimization problem in operations research. This problem involves deciding which pattern should be applied to raw material stock in order to obtain sufficient amount of products to meet the demands while minimizing cost. The CSP can be modeled and solved as an Integer Linear Program (ILP). In this paper we deal with a variant of CSP introduced in [7] that we classify [3] as $*/V/D/R$ using Dyckoff's typology, where $*$ means any dimensionality, V means that the raw material stock is sufficient to accommodate all the demanded products (hence, only some selected stock pieces have to be cut), D means that all large pieces are different (in terms of shape) and R indicates many products demanded of few different types. The feature V (any demand can be fulfilled) entails that the raw material stock to be cut has to be selected. Each large piece has an associated "value" (e.g. typically values are proportional to their sizes) and the objective function is to minimize the total value of the raw material stock selected to fulfill the demand.

According to a later typology presented in [8], we are dealing with a variant of the *Multiple Stock Size Cutting Stock Problem* (MSSCSP). In [8] it is noticed that research on cutting and packing problems still is rather traditionally oriented. For instance, few

recent papers consider heterogeneous assortments of large pieces. In addition, the variant analyzed in this paper has certain peculiarities that make it harder and consequently more challenging. This variant might emerge in real-life applications in which the number of raw material pieces available for cutting and their dimensions is uncertain because: (i) only a sample of the whole set of pieces is known, and/or (ii) the pieces might change dynamically with time. Population harvesting (e.g. plants, fish and animals [4]) are examples of both types of uncertainty: the dimensions of their raw elements might change due to their growth (case ii); and only some samples of the dimensions of the raw elements are taken by the industry (case i). Note that measuring all of them (there are possibly several hundred/thousand of elements) would be too costly.

Due to the above mentioned uncertainties associated with the raw material pieces of this type of MSSCSP, it is impossible to know all the patterns associated with each piece of stock (many dimensions of the stock are unknown). For this reason, in the literature and in industry, an algorithm that simulates the cutting of a whole set of raw material samples according to certain values vector has been generally used. The objective is to obtain similar results when such vectors and algorithms are used for cutting the real stock from which the sample data was acquired. The values vector is composed of continuous variables and each of them is associated with a product type. Note that we do not have access to the set of patterns to be cut in a direct manner, only via the application of this complex algorithm which is denoted throughout the paper as $\mathcal{A}$. The $\mathcal{A}$ algorithm selects the optimal cutting for each raw material sample based on the values of the products. The optimality criterion of such algorithm is to maximize the total value, which is the sum of the products of value and units cut of each product type. Then each combination of total amounts of products that can be cut from a set of raw material pieces represents a global products pattern for this set. By providing several different values vectors as input to the $\mathcal{A}$ algorithm, different global product patterns associated with a set of raw material pieces can be obtained. Once the best values vector has been selected in this cutting simulation process over the sample data, it is used as input to the $\mathcal{A}$ algorithm that cuts the real raw material (which is installed in the cutting machines). If the sample is representative of the whole population, the results of the cutting will be similar to that predicted in the simulation phase.

We would like to highlight that it is not straightforward to model and solve this variant of MSSCSP because the number of values vectors is infinite (continuous variables) and therefore, for realistic instances we can not enumerate all the global products patterns in a reasonable amount of time. For this reason a representative sample of them must be selected. This is a complicated task because the algorithm that generates the global products patterns (a) is complex and requires a great amount of time for realistic instances and (b) it matches many different values vectors to similar global products patterns. As an example of (b) consider a values vector whose associated global products pattern has the maximum possible amount of each type of product. Then, even if its associated value in the vector is increased (and the rest of values of the vector remain the same), the same global products pattern will be obtained. The latter fact, and the necessity of finding a representative set of global products patterns in a reasonable amount of time (case a), has motivated the work presented here.

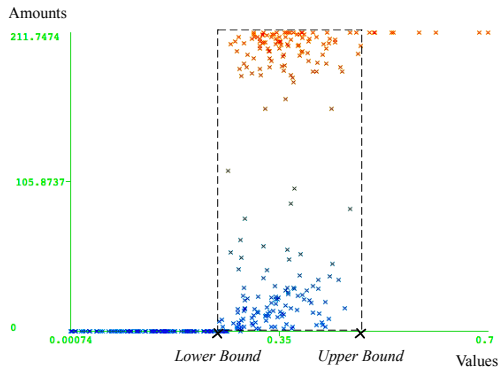

Fig. 1. Amounts obtained with $\mathcal{A}$ for several values associated with a product.

Our main objective is to reduce the search space for the values vector by bounding it in such a way that areas that produce the same global products pattern are excluded (because we want to maximize how scattered the global products patterns produced are). To illustrate this we show a graph in Figure 1 that represents the amount of certain products obtained after applying the $\mathcal{A}$ algorithm with different values vectors (other types of products can also be cut from the raw material). On the horizontal axis is the value associated with each type of product (normalized to the interval [0,1], and on the vertical axis are the product amounts. Note that the minimum amount that it is possible to obtain is zero units and the maximum amount is 211.74 units. The dashed rectangle includes different amounts of such products, so it is necessary to sample in this area in order to obtain a wide range of different global products patterns. Note that values less than or equal to the minimum value in the rectangle (lower bound), the $\mathcal{A}$ algorithm produces the same amount of product: zero. The opposite occurs with values that are at least the maximum value in the rectangle (upper bound): the amount obtained is the maximum. For this reason, using values that are outside the interval delimited by the lower and upper bound is a waste of time as no new global products patterns will be obtained. As mentioned, reducing the computational time is vital, especially in on-line problems such as CSP real-life applications with uncertainties.

In order to reduce the computational time for generating a representative sample of global products patterns, we present definitions and equations for calculating the lower and upper values bounds of each product. These bounds ensure that the minimum/maximum amount of a type of product is obtained. As far as we know, there is no other reported technique that attempts to compute the lower and upper bounds, nor to reduce the values search space. The lower and upper bounds that we introduce can be used with any sampling technique (e.g. random sampling as the naivest approach) and consequently by bounding the search space, a representative set of global products patterns tends to be obtained more quickly. In addition, without lose of generality, in this paper we also propose a method for exhaustive sampling using such bounds.

The paper is structured as follows. First, the variant of the MSSCSP is formalized. An explanation of the $\mathcal{A}$ algorithm is also provided. Afterwards, the calculation of the upper and lower bounds is explained, and we also introduce an exhaustive sampling algorithm. The effectiveness of our method is shown with an evaluation with real-life instances. Finally we present our conclusions.

2 Problem Formalization

In this section we explain the new features of the variant of the MSSCSP, with respect to the traditional CSP formulation [6]. (Parts of the following explanations have been extracted from [7]). MSSCSPs have raw material pieces of different dimensions which we can cut at will. In the variant that we are dealing with, we have a fixed number of raw material pieces (possibly hundreds or thousands) each with its own dimensions σ_r. There are K subsets of raw material pieces and each subset has R pieces (R might be different for each subset but this is ignored to simplify the description). Either a subset is fully cut with a unique values vector or none of its pieces is cut (as previously motivated in Section 1, due to the uncertainty of the environment). Then, each subset of raw material pieces has its own associated global products patterns (which are not given and therefore we must sample them), which are the combinations of amounts of products that can be cut from it. A global products pattern $\boldsymbol{p} \in {\mathbb{Q}_+}^{|\mathcal{M}|}$ is noted as $\boldsymbol{p} = \langle a_1, \ldots, a_{|\mathcal{M}|} \rangle$, where $\mathcal{M}$ is the set of product types and a_j represents the amount of units of product $m_j \in \mathcal{M}$ cut from certain set of raw material pieces.

Definition 1. *We represent a type of product as a tuple $m_j = \langle s_j, z_j \rangle$, where:*

- *$s_j \in \mathbb{R}$ is the size of a piece of m_j. Depending on the number of dimensions analyzed, s_j can represent: lengths for 1-dimension (e.g. cm), areas for 2-dimensions (e.g. cm^2) or volumes for 3-dimensions (e.g. cm^3).*
- *z_j is the dimensions of m_j. For instance, if m_j has the shape of a rectangle, z_j would be the required length and width for m_j.*

As previously mentioned, in the variant of MSSCSP analyzed, the patterns are not known *a priori* and it is only possible to have indirect control over them via a list of continuous variables called a values vector. A values vector $\boldsymbol{v} \in {\mathbb{Q}_+}^{|\mathcal{M}|}$ is a vector of $|\mathcal{M}|$ continuous variables. Each v_j represents the value associated with the type of product $m_j \in \mathcal{M}$ per unit of s_j. For instance v_j could represent monetary units: €, \$, etc. per each unit of s_j, (e.g. $€/m^3$). A set of products types $\mathcal{M}$ and a vector of dimensions σ of $|R|$ raw material pieces can be passed to an algorithm $\mathcal{A}$ which uses a values vector for calculating the corresponding $\boldsymbol{p}$. Then, $\mathcal{A}$ can be represented as the following mapping function:

$$\mathcal{A}(\mathcal{M}, \langle \sigma_1, \ldots, \sigma_{|R|} \rangle, v) \rightarrow p \tag{1}$$

To make this variant of the MSSCSP amenable to an ILP approach, in [7] the infinite set of possible values vector was reduced to a finite set of n values vectors (which should be sufficiently representative) $\boldsymbol{u}_{ik}$ $(i = 1 \ldots n)$ for each subset of raw material pieces

$k \in K$ (the same n is assumed for each subset to simplify the notation). Then global products patterns for each subset k are precomputed by using algorithm $\mathcal{A}$, storing the results in vectors of constants $\boldsymbol{u}_{ik} = \mathcal{A} : (\mathcal{M}, \langle \sigma_1, \ldots, \sigma_{|R|} \rangle_k, \boldsymbol{v}_{ik})$ $(\forall i, k)$. The ILP model is as follows:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{k=1}^{K} c_k x_{ik} && \forall x_{ik} \in \{0,1\} \\
\text{s.t.} \quad & \sum_{i=1}^{n} \sum_{k=1}^{K} u_{ikj} x_{ik} \geq d_j && \forall j \in \mathcal{M} \\
& \sum_{i=1}^{n} x_{ik} \leq 1 && \forall k \in K
\end{aligned}
$$

where d_j is the targeted demand for each type of product, c_k is the value associated with the stock subset k and x_{ik} are the decision variables that indicate if the subset k is cut with the global product pattern i. The objective function is to minimize the total value of the sets of raw materials used for satisfying the demands. Note that if a subset of raw material pieces k is not used for satisfying the demands (and therefore it is not cut), then all its decision variables ($x_{ik} \forall i \in n_k$) are zero. Note also that the first constraint ensures that the demands are fulfilled and second constraint prevents the use of more than one global products pattern in a set of raw material pieces. As mentioned, this set of representative global products patterns ($\boldsymbol{u_{ik}} \forall i \forall k$) must be generated in order to be able to solve this ILP with standard optimisation software; and it needs to adequately cover all possible global products patterns for each set of raw material pieces k. The main contribution of this paper resides in this task. By bounding the search space of values of the vector, we are reducing the likelihood of generating global products patterns that are not new (they are equal to a previous generated global products pattern). Hence, the global products patterns generated in a fixed amount of time tend to be more scattered and therefore more significant for the analyzed problem.

3 Algorithm $\mathcal{A}$

In [7], the $\mathcal{A}$ algorithm is treated as a black box. Instead, we analyze and use its properties, which allows us to reduce the values search space of the this variant of the MSS-CSP. For this reason, in this section we briefly explain the $\mathcal{A}$ algorithm. This algorithm simulates the cutting of a set of raw material pieces R into certain products types $\mathcal{M}$. Each type of product m_i has an associated value v_i, representing how valuable is each unit of product (these values compose the values vector v, which is provided as an input). This algorithm selects the optimal cutting for each raw material sample, where the optimality criterion is to maximize the total value, which is the sum of the products of value and units cut of each product type.

Definition 2. *The total value of a piece of product $m_j \in \mathcal{M}$ is calculated as:*

$$t(m_j) = s_j v_j \tag{2}$$

Note that the input values vector has a direct impact on the amounts of each type of product that will be obtained from a certain raw material. The other factor that has an influence in the amounts is the dimensions of the raw material pieces and the dimensions of the product types. The greater is a value v_i where the other values of v are fixed, the greater the number of products m_i its associated global products pattern p will have after running algorithm $\mathcal{A}$ (they will be equal only in the case of reaching a saturation point, see Figure 1 as an example). In the same way, the opposite situation (at most number of pieces) holds when the values of v_i are decreased. In the next section, we use these properties of the $\mathcal{A}$ algorithm for bounding the values search space.

Typically, in the literature, the $\mathcal{A}$ algorithm has been implemented with Dynamic Programing (DP) [2, 1]). DP is an approach that allows us to solve complex problems by dividing them into a collection of simpler subproblems. For such purpose, the subproblems must be sequential and independent. The problem of cutting a raw material piece satisfies these properties, since it is a recursive one (i.e. maximize by cutting the first product and then maximizing the remainder).

4 Computing Lower and Upper Bounds

In this section we present definitions and equations of the lower and upper values vector that is provided as an input of the $\mathcal{A}$ algorithm. Each value v_i of such a vector (v) is associated with a type of product m_i with particular shape characteristics z_i. The main idea behind the lower and upper bounds is that when we apply the $\mathcal{A}$ algorithm (Equation 1) to a certain raw material, when the value associated with a type of product is:

(i) at most the lower bound, it is ensured that the amount obtained of such product is the minimum.
(ii) at least the upper bound, it is ensured that the amount obtained of such product is the maximum.

A toy example is described for further explanation of the bounds.

Example 1. We consider a 2-dimensional space with two product types (m_1 and m_2) with a rectangle shape. Then z_1 and z_2 is represented with the height (h) and the width (w). The size of the product types, is determined by the equation of the area of a rectangle. Thus, $s_1 = l_1 w_1$ and $s_2 = l_2 w_2$. Figure 2 shows such products. Note that s_1 is greater than s_2. Without loss of generality, for this example, $w_1 = w_2$.

4.1 Computing Lower Bounds

Given $\mathcal{M} = \{m_i, m_j\}$ where $s_j < s_i$ (in Example 1, $m_i = m_1$ and $m_j = m_2$), we want to calculate the lower value bound, which is the greatest value that we can assign to the associated variable of m_i in a values vector (v) in order to ensure that the minimum amount of m_i is obtained for any raw material. For such purpose, we present the following definition.

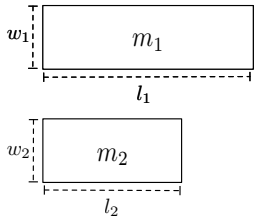

Fig. 2. 2- dimensional example of two product types.

Definition 3. *The maximum number of pieces of m_j that can fit in one piece of m_i, according to their shape specifications (z_i and z_j), is $k(z_i, z_j)$. To simplify the notation, we will tend to use k rather than $k(z_i, z_j)$ when the variables are obvious from context.*

We consider a subpart of the raw material from which we can only cut either one piece of m_i or k pieces of m_j. Note that the waste produced when cutting m_i is *smaller or equal* than when cutting k pieces of m_j. Figure 3 shows such situation for Example 1, where the size of the subpart of the raw material analyzed is ks_1. Note that for this example, $k(z_1, z_2) = 1$. If the cut off products from this subpart is k pieces of m_2, there is an associated waste which is the size analyzed minus ks_2 (grey area in Figure 3). Thus, if $v_1 = v_2$, cutting the type of product m_1 is the best option because the total value of a piece of m_1 is greater than the total value of k pieces of m_2: $t(m_1) > kt(m_2)$ due to $s_1 \geq ks_2$ (see Equation 2).

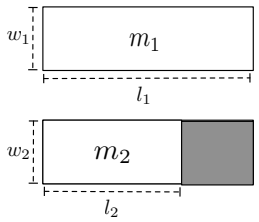

Fig. 3. Example for explaining the lower bound.

For computing the lower value bound, we would like to know how much smaller the value v_i should be, in order to change the priority order where cutting k pieces of m_j is the most profitable. In order to compute this value, we compute for which v_i the priorities are equal. This situation occurs when the total profit of cutting a piece of m_i is equal to the total profit of cutting k pieces of m_j ($t(m_i) = kt(m_j)$). By applying Equation 2 we obtain for which v_i the priorities are equal. Then, resting an arbitrarily small positive number (denoted as ϵ), we obtain the lower bound:

$$v_i^{lb}(m_i, m_j) = \frac{k(z_i, z_j)s_j v_j}{s_i} - \epsilon, \text{ for } s_i \geq s_j. \tag{3}$$

From the above, we can state that for any value $v_i \leq v_i^{lb}$, the total benefit of cutting k pieces of m_j is greater than the total benefit of cutting a piece of m_i. Therefore, when applying the $\mathcal{A}$ algorithm with v_i^{lb} to a complete raw material, the obtained number of pieces of m_i is the minimum and the obtained number of pieces of m_j is the maximum. Note that if at least one piece of m_j fits in a piece of m_i (e.g. Example 1 represented in Figure 2), then the minimum amount of m_i is zero (e.g. Figure 1). Otherwise, there might exist subparts of the raw material in which m_j does not fit but m_i does, in such case the amount of m_i could be greater than zero.

4.2 Computing Upper Bounds

In this section, given the same $\mathcal{M} = \{m_i, m_j\}$ where $s_j < s_i$ (in Example 1, $m_i = m_1$ and $m_j = m_2$), we want to calculate the upper value bound, which is the lowest value that we can assign to the associated variable of m_i in a values vector (v) in order to ensure that the maximum amount of m_i is obtained for any raw material. For such purpose, we define:

Definition 4. *The minimum number of pieces of m_j that are required to fit one piece of m_i into the global shape of them, according to their shape specifications (z_i and z_j), is $h(z_i, z_j)$. To simplify the notation, we will tend to use h rather than $h(z_i, z_j)$ when the variables are obvious from context.*

We consider a subpart of the raw material from which we can only cut either one piece of m_i or h pieces of m_j. Note that the waste produced when cutting m_i is *at least* that when cutting h pieces of m_j. Figure 4 shows such situation for Example 1, where the size of the subpart of the raw material analyzed is hs_2. Note that for this example, $h(z_1, z_2) = 2$. If the cut off product from this subpart is m_1, there is an associated waste which is the size analyzed minus s_1 (grey area in Figure 4). Thus, if $v_1 = v_2$, cutting the type of product m_1 is the worst option because the sum of the total value of the h pieces of m_2 is greater than the total value of a piece of m_1: $t(m_1) < ht(m_2)$ due to $s_1 \leq hs_2$ (see Equation 2).

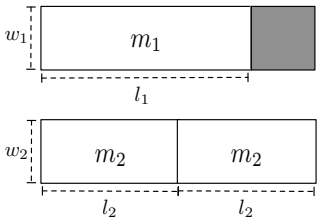

Fig. 4. Example for explaining the upper bound.

For computing the upper value bound, we would like to know how much greater the value v_i should be, in order to change such priority order (a piece of m_i the most profitable). In order to compute such a value, as we did previously, we compute for

which v_i the priorities are equal. This situation occurs when the total profit of cutting a piece of m_i is equal to the total profit of cutting h pieces of m_j ($t(m_i) = ht(m_j)$). By applying Equation 2 we obtain for which v_i the priorities are equal. Then, summing an arbitrarily small positive number (denoted as ϵ), we obtain the upper bound:

$$v_i^{ub}(m_i, m_j) = \frac{h(z_i, z_j)s_j v_j}{s_i} + \epsilon, \text{ for } s_i > s_j. \tag{4}$$

From the above, we can state that for any value $v_i \geq v_i^{ub}$, the total benefit of cutting h pieces of m_j is lower than the total benefit of cutting a piece of m_i. Therefore, when applying the $\mathcal{A}$ algorithm with $v_i \geq v_i^{ub}$ to a complete raw material, the obtained number of pieces of m_i is the maximum and the obtained number of pieces of m_j is the minimum. Because when it is possible to cut from a certain subpart of the material h pieces of m_j, instead, a piece of m_i will be cut. Note that the minimum amount of m_j does not have necessarily (and probably will not) to be zero. This is due to the fact that $s_j < s_i$ and therefore, there will probably be parts of the raw material in which m_i does not fit but m_j does.

4.3 Generalizing the Bounds

Previously we introduced the equations of the lower and upper bounds that ensure that we obtain the minimum and maximum amounts of a type of product in comparison with a smaller type of product. Here, we extend these concepts for the circumstance in which there are more than two types of products to be cut. First, we denote the smaller subset of a product type m_i as:

Definition 5. $\mathcal{M}_i^{<} \subset \mathcal{M} : s_j < s_i, \forall m_j \in \mathcal{M}$.

We present two propositions:

(i) If v_i is equal to the minimum value of all the lower bounds associated with each smaller type of product, we can ensure that the minimum amount of m_i will be obtained, since the total value of cutting k pieces (see Definition 3) of any of the smaller products is greater than the total value of cutting a piece of m_i. This is denoted as follows:

$$v_i^{lb}(m_i, \mathcal{M}_i^{<}) = \min_{m_z \in \mathcal{M}_i^{<}} v_i^{lb}(m_i, m_z). \tag{5}$$

(ii) If v_i is equal to the maximum value of all the upper bounds associated with each smaller type of product, we can ensure that the maximum amount of m_i will be obtained in the homogeneous case (combinations of products of the same type only), since the total value of cutting h homogeneous pieces of any other smaller products is lower than the total value of cutting a piece of m_i (see Definition 4). This is denoted as follows:

$$v_i^{ub}(m_i, \mathcal{M}_i^{<}) = \max_{m_z \in \mathcal{M}_i^{<}} v_i^{ub}(m_i, m_z). \tag{6}$$

For the case of combinations of h heterogeneous pieces (combinations of products of different types) of smaller products, unfortunately it is possible that some heterogeneous combinations have a *slightly* greater total value than a unit of product m_i. Then, we cannot assume Proposition (ii) for all the heterogeneous combinations. However, it is very unlikely that such proposition does not hold for real-life instances. This is because when we compute the bounds, we consider that the space left in the area analyzed is waste (worst scenario). However, in reality such left area could be used to fit another piece of any product (generally several pieces fit in every big raw material piece). Then the total value of h heterogeneous pieces (excluding m_i) would be compared against the total value of n heterogeneous pieces (including at least a piece of m_i). (However in our proposition only a unique piece of m_i is considered, which has lower total value than combining such piece with another product). For this reason, generally for real instances the min./max. amounts of products are obtained with greater/lower values (respectively) than the theoretical bounds presented in this paper.

We now define the interval of values between the lower and upper bounds of a type of product with respect smaller types of products. This allows us a reduction of the values search space while ensuring that global products patterns with amounts between the minimum and maximum possible amounts (inclusive) are selected (according to Propositions (i) and (ii)). Such a set of values is defined as follows:

$$\mathcal{V}_i(m_i, \mathcal{M}_i^<) = [v_i^{lb}(m_i, \mathcal{M}_i^<), v_i^{ub}(m_i, \mathcal{M}_i^<)] \tag{7}$$

In [7] the authors generate a set of global products patterns with Monte Carlo simulation over a fixed interval for all the types of products (e.g. $[1, 1000]$). Instead, this simulation can be performed over $\mathcal{V}_i(m_i, \mathcal{M}_i^<)$ by fixing a basis value for the smallest product and computing the $\mathcal{V}$ interval for the bigger products (in increasing size order) for each sampling. As mentioned, by calculating the specific interval delimited by the lower and upper bounds for each type of product, we are reducing the likelihood of generating equal global products patterns, which implies a greater likelihood of obtaining scattered global products patterns. Following, we also introduce an exhaustive global products patterns generation algorithm that uses such intervals.

5 Exhaustive Global Products Patterns Generation based on $\mathcal{V}$

In this section we explain how to generate all possible global products patterns for a set of product types with respect to some fidelity (denoted as f) over the values vector. First the $\mathcal{V}$ interval is discretized based on f (see Equation 7), then we present an algorithm that exhaustively generates global products patterns.

5.1 A Fidelity-based Discretization of the Interval $\mathcal{V}$

Following the interval $\mathcal{V}$ (Equation 7) is discretized according to a fidelity variable f, which represents a value increment:

$$\begin{aligned}\mathcal{V}_i(m_i, \mathcal{M}_i^<, f) = &\{v_i^{lb}(m_i, \mathcal{M}_i^<) + nf, \forall n \in \{0, \ldots, q\}\} \\ &\text{where } q = \min \mathbb{N} : v_i^{lb}(m_i, \mathcal{M}_i^<) + qf \geq v_i^{ub}(m_i, \mathcal{M}_i^<)\end{aligned} \tag{8}$$

The above set of values is expressed as a minimum value and a series of increments of value f over it. The minimum value of the set is the lower bound. The next values are obtained by incrementing f units in every step. The maximum number of such increments is denoted as q and it is the minimum natural number of increments of value f that are necessary in order to reach or exceed the upper bound. Note that the lower the fidelity is, the greater the set $\mathcal{V}_i$ is (with the exception of rare situations in which the lower and upper bounds are equal).

5.2 Algorithm for an Exhaustive Generation of Global Products Patterns

We introduce an algorithm that generates all the values vectors for a set of product types $\mathcal{M}$ by computing the discretized set $\mathcal{V}$ (see Equation 8) for a given fidelity f. The corresponding global products patterns (denoted as C) of the values vectors are also computed by using the $\mathcal{A}$ algorithm over a given subset of raw material pieces with characteristics $\langle \sigma_1, \ldots, \sigma_{|R|} \rangle$. First, Algorithm 1 initializes an empty values vector. Then, it assigns a basis number (denoted as b) to the smallest type of product, where b can be randomly generated or it can be specified by the user. Note that this value remains fixed during the complete execution of the algorithm. Algorithm 1 is also in charge of initializing the subset $\mathcal{M}_u \subset \mathcal{M}$, which contains the products whose values have not yet been assigned (at this stage, all the products except the smallest one).

Algorithm 1: Exhaustive Generation of Global Products Patterns

Data: $\langle \sigma_1, \ldots, \sigma_{|R|} \rangle, \mathcal{M}, f, b$
Result: C
$m_i \in \mathcal{M} : s_i = \min_{z \in |\mathcal{M}|} s_z$;
$\boldsymbol{v} \leftarrow \emptyset$; // empty values vector
$v_i \leftarrow b$;
$\mathcal{M}_u \leftarrow \mathcal{M} \backslash \{m_i\}$; // Unassigned set of products
$C \leftarrow$ `fixValue` $(\boldsymbol{v}, \boldsymbol{\sigma}, \emptyset, \mathcal{M}, \mathcal{M}_u, f)$;
return C

Finally, Algorithm 1 calls the recursive procedure `fixValue` which, given the set of unassigned product types ($\mathcal{M}_u$), selects the smallest one and computes its set of values ($\mathcal{V}$, see Equation 8). Subsequently, each of the values of such a set is assigned iteratively to the analyzed type of product. In addition, the already assigned type of product is deleted from $\mathcal{M}_u$. This process is repeated recursively until all the product types have already an assigned value. Note that the procedure `fixValue` is recursively called with the updated set $\mathcal{M}_u$. Once all the product types have already an assigned value, the $\mathcal{A}$ algorithm is used for obtaining the global products pattern associated with the vector of values v. If such global products pattern is new, it is added into C. Finally, when all the values in the set $\mathcal{V}$ have been assigned, the procedure adds the type of product to the unassigned set of products ($\mathcal{M}_u$) and it returns the set of global products patterns computed C. Once all the runs of the procedure `fixValue` have finished, the

total set of global products patterns generated is returned to the Algorithm 1. (Note that all the patterns obtained by each call to the procedure are merged into C).

Procedure fixValue($\boldsymbol{v}, \boldsymbol{\sigma}, C, \mathcal{M}, \mathcal{M}_u$, f) : $\mathcal{C}$

```
select m_i ∈ M_u : s_i = min_{z∈|M_u|} s_z ;
M_u ← M_u \ {m_i};
V ← V_i(m_i, M_i^<, f); // See Equation 8
for e ∈ V do
    v_i ← e;
    if M_u = ∅ then
        ⟨a_1, ..., a_|M|⟩ ← A(M, ⟨σ_1, ..., σ_|R|⟩, v);
        if ⟨a_1, ..., a_|M|⟩ ∉ C then
            C ← C ∪ {⟨a_1, ..., a_|M|⟩};
    else
        C ← C∪ fixValue (v, σ, C, M, M_u, f);
M_u ← M_u ∪ {m_i};
return C
```

6 Evaluation

In this section we first describe a case study of a real-world population harvesting problem [4]: forestry harvesting. Subsequently we evaluate several instances of this type with our approach. We do not compare our solution with other techniques because, as far as we know, there is no other approach that attempts to compute such bounds, nor to reduce the values search space. In the forestry harvesting problem the logs of the trees have to be cut into smaller log-pieces by harvesting machines in order to satisfy the demands of the customers. The products have shape specifications (minimum diameter, length, etc.) and their volumes are measured typically in cubic meters (m^3). As mentioned, in this variant of the MSSCSP there are several subsets of stock. For the forestry problem, we call each subset of trees a *block* and each of them has a given specific value. The objective function is to minimize the total value associated with the blocks harvested. (Recall that for the MSSCSP only some stock is selected for satisfying the demands and therefore only this selection is harvested).

In this section we evaluate the approach presented in this paper, which reduces the values search space by computing upper and lower values bounds associated with all the products. For such purpose, we used the exhaustive sampling approach introduced in this paper (Algorithm 1) for generating a set of global products patterns for each block. Lastly, by solving the ILP of the MSSCSP variant analyzed (see Section 2), we obtain the optimal solution for such patterns. A solution of this variant of the MSSCSP consists in a selected vector of values (and its corresponding global products pattern) for each of the selected blocks to be cut. Note that there is no guarantee that this solution is the optimal based on all the possible global products patterns. Due to the impossibility of

enumerating all the possible global products patterns for realistic instances, optimality can not be ensured for the analyzed variant of the MSSCSP. Hence the importance of obtaining a sparse (and therefore significant) sample of global products patterns.

We have performed the evaluation with a sample of real data from our industrial partner. The total volume of the sampled logs of the trees is $1191.3m^3$ and it is composed by eight blocks and four product types. We computed the global products patterns of each block by using Algorithm 1 with $b = 10$ and fidelities: 0.9, 0.7 and 0.5. Then, we solved 50 randomly generated and satisfiable demands instances by solving their corresponding ILPs with CPLEX solver with a time cut-off of 1 hour (but the average of the solving time was 16 minutes). The experiments were run on a 2.3 GHz Intel Core $i7$ processor. Our industrial partner also provided us with the software that carries out the DP-based simulation that implements $\mathcal{A}$ (see Equation 1). Since DP is a complete algorithm, other DP implementations, such as the ones mentioned in Section 3, could have been used for $\mathcal{A}$, with equivalent results.

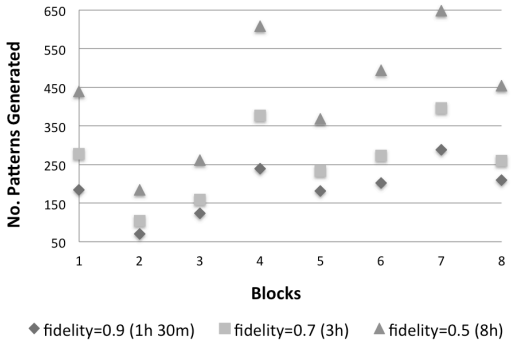

Fig. 5. Number of global products patterns generated for each block.

Figures 5 and 6 show the results obtained from the experiments performed. Specifically, Figure 5 shows the number of global products patterns that Algorithm 1 computed for the tested fidelities. As mentioned, the lower the fidelity the higher the number of global products patterns, in general. For instance in Block 4 and Block 7 there is a difference of almost 400 global products patterns between fidelity 0.5 and 0.9. This is reflected in the computation times of the global products patterns (see the times between brackets in Figure 5). Note that computing the global products patterns for all the blocks for fidelity 0.9 required one hour and a half, which is less than five times less the time required for fidelity 0.5 (eight hours). However, our algorithm allows the selection of the global products patterns granularity according to the available time.

The differences in the number of global products patterns computed among the different blocks depend on the characteristics of the blocks (such as number of pieces of raw material and their sizes). Note that it is more likely to generate equal global products patterns (which are rejected by Algorithm 1) for smaller block sizes. For our

analyzed instance, Block 2 is the smallest block (in terms of total m^3) and it has the lowest number of generated global products patterns (see Figure 5).

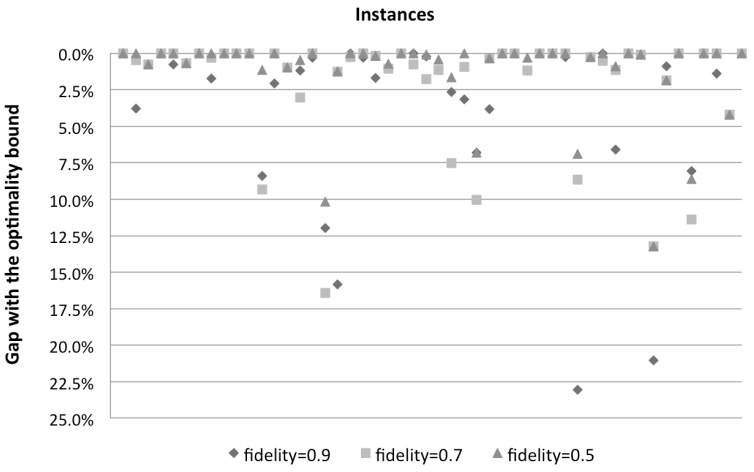

Fig. 6. Quality of the solutions obtained.

We compute an optimality bound by obviating the global products patterns and considering that any combination of amounts of products can be cut from each block (within its size). (It can be obtained by making such variations over the ILP of Section 2). Note that it is a bound of the optimality because at least such a total value must be expended for satisfying certain demands. However, it might occur that the global products patterns of the optimality bound do not exist in reality. In such a case, the optimal solution has a greater total value than the optimality bound.

Figure 6 shows the quality of the solutions obtained after solving the ILPs of the 50 random demands instances, which is expressed as a gap with the optimality bound (left axis). Note that for 41 instances we obtained a gap lower than 5%, which can be considered as near-optimal. In addition, for many of them the gap is 0%, which means that they are optimal. As expected, with better (lower) fidelities (which very often implies more global products patterns) the quality of the solutions are equal or better. Needless to mention, the economical impact that the quality of these solutions has in the real-life applications (for real instances, the value of a single block could possible be several thousand €). In addition to the sampling performed with our approach, a complementary subsequent clustering algorithm (such as the one presented in [7]) could be applied in order to reduce the set of input global products patterns provided to the ILP (with the objective of speed up its solving time).

7 Conclusions

In this paper we have contributed to the literature by introducing an approach that bounds the search space of a variant of the MSSCSP that arises from population harvesting. For such a problem it is not possible to enumerate all the patterns in a reasonable amount of time, and therefore it is necessary to find a sparse set of global product patterns. In this paper we provide definitions and equations of the lower and upper values bounds of the products. By sampling in their interval, the likelihood of generating equal global products patterns is lower (and therefore patterns tend to be more scattered). Furthermore, we also have introduced an algorithm that exhaustively generates global products patterns according to their bounds and a fidelity parameter that fixes their granularity.

The evaluation performed with a harvesting problem from our industrial partner showed that the better the fidelity, the more global products patterns are generated, and the better the quality of the solutions tends be. Most of the solutions obtained were near-optimal or optimal, specially for the best fidelity analyzed. We would like to highlight that the number of global products patterns, and how representative they are, affects the quality of the solutions, so it has an economic impact for the stock owners.

As future work, we will focus on applying this approach to other types of real-life MSSCSPs that fit into the population harvesting framework (e.g. we found similarities with a problem from the clothing industry [5]).

Acknowledgments

This research was supported in part by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

References

1. D. Anderson, D. Sweeney, T. Williams, J. Camm, and J. Cochran. *An introduction to management science: quantitative approaches to decision making*. Cengage Learning, 2015.
2. R. Bellman and S. Dreyfus. *Applied Dynamic Programming*. Princeton, 1962.
3. H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159, 1990.
4. W. M. Getz and R. G. Haight. *Population harvesting: demographic models of fish, forest, and animal resources*, volume 27. Princeton University Press, 1989.
5. M. Gradišar, M. Kljacić, G. Resinovič, and J. Jesenko. A sequential heuristic procedure for one-dimensional cutting. *European Journal of Operational Research*, 114:557–568, 1999.
6. L. V. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, 6(4):366–422, 1960.
7. S. D. Prestwich, A. O. Fajemisin, L. Climent, and B. O'Sullivan. Solving a hard cutting stock problem by machine learning and optimisation. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2015.
8. G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.